GCCS/DII COE System Integration Support

Scientific and Technical Reports: DII COE Studies and Analysis (Software Quality Compliance Plan for DII COE 3.0)

January 22, 1997

Prepared for:

DISA/JEJA
ATTN: Claire Burchell
45335 Vintage Park Plaza
Sterling, VA 20166-6701

Contract Number: DCA100-94-D-0014
Delivery Order Number: 330, Task 6
CDRL Number: A029

Prepared by:

Computer Sciences Corporation
Defense Enterprise Integration Services
Four Skyline Place
5113 Leesburg Pike, Suite 700
Falls Church, VA 22041

THIS DOCUMENT IS UNCLASSIFIED

Defense Information Infrastructure (DII)

Common Operating Environment (COE)

Software Quality Compliance Plan for DII COE

Version 3.0

January 22, 1997

Prepared for:

DISA/JEJA
ATTN: Claire Burchell
45335 Vintage Park Plaza
Sterling, VA 20166-6701

Prepared by:

Computer Sciences Corporation
Defense Enterprise Integration Services
Four Skyline Place
5113 Leesburg Pike, Suite 700
Falls Church, VA 22041

Table of Contents

List of Figures

List of Tables

Preface

The following conventions are used in this document:

Bold	Used for information that is typed, pressed, or selected in executables and instructions. For example, select **connect to host**.
Italics	Used for file names, directories, scripts, commands, user IDs, document names, and Bibliography references; and any unusual computerese the first time it is used in text.
Underline	Used for emphasis.
Arrows <>	Used to identify keys on the keyboard. For example <Return>.
"Quotation Marks"	Used to identify informal, computer-generated queries and reports, or coined names; and to clarify a term when it appears for the first time. For example "Data-Generation Report."
`Courier Font`	Used to denote anything as it appears on the screen or command lines. For example `tar xvf dev/rmt/3mm`
Capitalization	Used to identify keys, screen icons, screen buttons, field, and menu names.

This page intentionally left blank.

1. INTRODUCTION

1.1 Purpose

Metrics analysis of software development products provides a non-intrusive process to identify suspect medium-to-high risk components. The visibility gained from the resulting information can be used proactively as an effective management technique to control the risks of integrating third party software.

The essential value of metrics was elegantly expressed by the highly respected English chemist Lord Kelvin, circa 1889, who stated at a lecture before the Royal Society:

> When you can measure what you are speaking about and express it in numbers, then you know something about it; but when you cannot measure, when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind: but you have scarcely, in our thoughts advanced to the stage of a science.

The contents of the Defense Information Infrastructure (DII) Software Quality Compliance Plan provides a description of the metric collection, analysis activities, and schedule. These collection and analysis activities evaluate the level of compliance to the Common Operating Environment (COE) software quality goals. The value of the compliance level assists the DII Engineering Office in understanding the costs and risks of integrating software into the COE to provide common functions. This understanding is used to determine the level of support needed for each COE common function. Specifically, this document describes the metrics collection and analysis process that will ensure the compliance to the software quality guidelines required to provide functionality for the COE.

The degree and ease of "plug and play" open systems integration is highly dependent upon compliance with the processes contained in the DII COE Integration and Runtime Specification (I&RTS). An integral part of this compliance is the operational risk associated with each function in the COE. The COE foundation for building open systems places the responsibility for most of the integration activities on the developers to ease the burden of source code level integration. A side effect of delegating this responsibility is that the COE is vulnerable to the unintentional introduction of software with residual faults.

The metrics collection and analysis process provides measurable values of software quality levels to assess the risk and level of support associated with software prior to becoming part of the functions provided by the COE. These software quality levels are determined by a series of automated, non-intrusive risk assessment processes.

The primary assessment process is the population analysis of software components that provide COE functions. The informational value gained from the identification process is twofold:

- Understanding the level of support for the medium-to-high risk functions.

- Tracking the variations in the medium-to-high risk population as versions of the software in the COE change.

Many software quality assessment techniques and metrics contributed from both the Government and private sector are described in the literature. The majority of these techniques are oriented from a specific developer's goals and objectives. A mature development organization has access to other types of supporting information to assess the degree of success in achieving an institution's Software Quality Assurance (SQA) goals using a specific metric set. An example of such a metric is the subjective evaluation of the adequacy of the comments. Currently, the DII does not have the information and resources required to perform this assessment for each of the COE candidates. Another example of such a metric set is the cost and schedule information.

The process and techniques contained in this plan provide the DII Engineering Office with a process that is objective, independent, repeatable, automated, and economical. The intent of this process is to provide insight into the risks associated with COE candidate software. The data provided by the process described in this plan is only part of the information used in the DII COE candidate evaluation process.

1.2 Scope of Plan

The activities and processes described in this document provide an independent, non-intrusive, and multi-factor assessment process to determine the level of support associated with COE Software Quality compliance. The contents of this plan describe the processes and information products resulting from:

- Analysis of component risk populations in applications.
- Complexity and quality analysis.
- Trend analysis of risk populations, complexity, and quality metrics.
- Compliance assessment of the COE Application Program Interfaces (APIs).
- Application integration testing effectiveness analysis.
- Application portability analysis.

The COE Software Quality compliance assessment process, depicted in Figure 1-1, uses automated, non-intrusive techniques implemented with commercial off-the-shelf (COTS) analysis tools. Specifically, the compliance assessment information is obtained by performing the metrics collection and analysis tasks described in this document.

The information gained from the complexity metrics and the resulting risk population analysis, Quality Criteria Profile (QCP), helps in both determining the level of support needed to integrate a function into the COE and how problematic the integration task will be. Individually, this information provides a small facet of a COE function candidate's risk assessment. Combining the QCP risk interval populations with metrics that assess compliance with COE APIs, code portability, and robustness data provides the DII Engineering Office a realistic forecast technique of the effort and associated risk of a candidate COE function.

The data obtained from the static metrics analysis provides the DII Engineering Office information on the overall complexity and quality assessment of the applications by generating the populations of the risk intervals for the QCP.

By focusing on a complexity and quality analysis of the design, implementation, and maturity of the development organizations, management information describing many common integration cost drivers (e.g., difficulty of changing the software, testing adequacy, and ease of portability to new platforms and updated versions of COTS) becomes available. Using additional analytic techniques, information projecting future integration difficulties by examining the potential "avalanche" of the current low-to-medium risk components into high risk population provides the DII Engineering Office with a glimpse into the level of support needed for future integration activities.

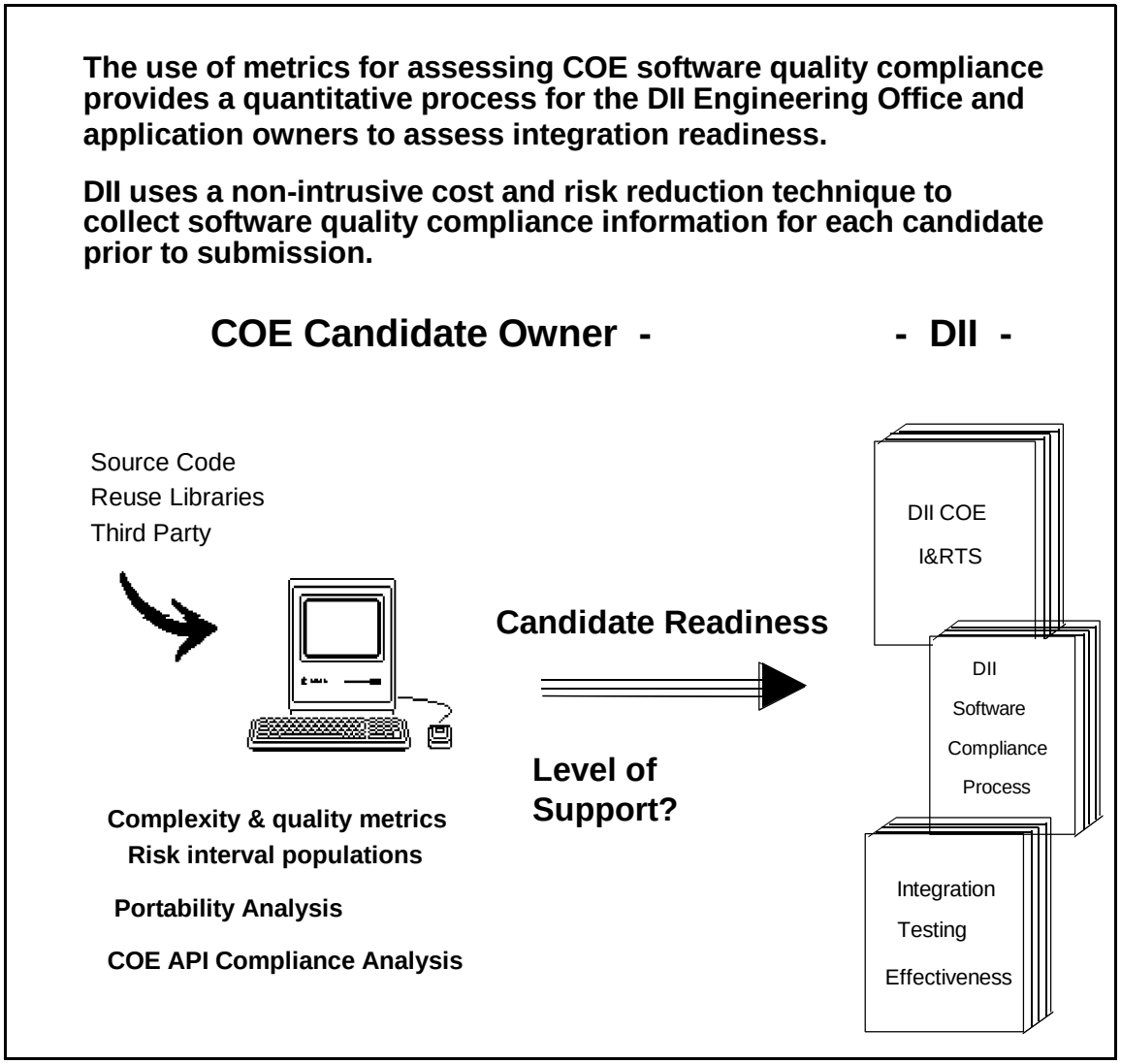

Figure 1-1. COE Software Quality Compliance Assessment

The information obtained from the portability metrics analysis assesses the degree of compliance with the approved COE configurations.

The values of these compliance metrics, when combined with the information in QCP risk populations, are helpful in projecting the effort associated with rehosting to different platforms. The portability analysis, when coupled with other types of COE programmatic information, support DISA's decision making process and planning for transfers of new hardware and software technologies.

1.3 Referenced Documents

Defense Information Infrastructure (DII) Common Operating Environment (COE) Integration and Runtime Specification (I&RTS), Version 2.0, October 23, 1995.

2. SOFTWARE QUALITY COMPLIANCE PROCESS

2.1 Software Quality Compliance Process Goals

The software quality compliance process described in this Plan is to provide risk assessment information to the DII COE Engineering Office to support the GCCS and GCSS activities of building and integrating systems by:

- Identifying suspect components in the candidates that present significant risk factors in current integration tasks; and problems in achieving and maintaining acceptable levels of maintainability, correctness, and reliability.

- Identifying components in COE functions that are cost effective candidates for renovation by tracking increases in medium-to-high risk populations and growth in complexity.

- Institutionalizing software quality compliance assessment techniques within the DII community to effectively manage costs and risks of integration by providing a COE software metrics and risk population database.

- Identifying usage of non-public APIs.

- Increasing testing effectiveness by analyzing data from Software Problem Reports (SPRs) and QCP risk population intervals.

- Identifying components in candidates with portability risks by detecting usage of operating system dependencies and non-compliance with approved COE COTS.

The software quality compliance assessment tasks produce information addressing each of these COE management concerns. The complexity and quality metrics are collected from each candidate's source code by using commercial static analysis tools. A number of metrics collection and analysis tools were used to provide the metrics in the dictionary. These tools collect metrics for a wide variety of languages including, but not limited to, FORTRAN, Ada, C, C++, Pro-C and FORTRAN, BAL, and PL-I. Currently, SQL is not supported by the vendors. The formulas for the QCP (see Appendix C) are based on this set of metrics.

These metrics are used to provide an empirical description of the risk populations for the three criteria in the QCP (Maintainability, Correctness, and Reliability), described in Table 2-1.

Table 2-1. QCP Definitions

QCP Criteria	Definition
Maintainability	Assessment of ease or difficulty in modifying the code.
Correctness	Extent of simplicity and structure of the logic in the application software.
Reliability	Assessment of the fragility of the code during execution.

The QCP information identifies current medium-to-high risk components, such as FORTRAN and C functions, as well as components in the software representing future risk as a result of continued software related activities by the development organizations. The future medium-to-high risk components represent excellent candidates for renovation as part of the ongoing incremental development activities to continually reduce COE risks and costs.

The collected metrics provide the DII Engineering Office with information empirically describing the effectiveness of the integration testing. Specifically, identifying components in the medium-to-high risk QCP populations pinpoints those components with a high likelihood of containing residual software faults. Discovery of these undetected faults, often embedded in complicated control structures, typically plague integration testing activities, increases the planned effort, and causes schedule slippage. Increased integration testing focuses on further validation of these identified components to enhance a candidate's readiness for COE releases.

The information obtained from portability analysis will assist the DII Engineering Office in making technology infusion decisions. Specifically, the QCP medium-to-high risk populations information can be combined with applications portability assessment metrics to support technology infusion planning for the DII. Applications identified as containing medium-to-high percentages of machine or Operating System (OS)-dependent code, usage of non-approved COE COTS, as well as large populations of medium-to-high risk components are significant obstacles in planning versions of COE using new technologies, such as CORBA.

2.2 Software Quality Compliance Information Collection

The DII focus of collecting information that describes different software quality characteristics of source code is to gain a better understanding of risk factors when planning COE releases. The DII software quality compliance evaluation currently contains five automated information collection activities to provide visibility into the integration planning process. The information is collected using the compliance checklists contained in Appendix A.

For each candidate, the information collection activities are:

- Static analysis of source code.
- Conformance analysis of APIs.
- Trend analysis of complexity and quality metrics.

- Integration test effectiveness analysis.
- Application portability analysis.

2.3 Static Analysis of Source Code

The static analysis of the candidate's source code builds a baseline of metrics containing information on the complexity and quality of the software. Additionally, these metrics provide the baseline to which future releases of a candidate can be compared. This is done to determine the status of the quality as modifications are being made. This information also pinpoints those components that may require further attention regarding testing; documentation; and, in some cases, redesign and reimplementation to reduce the complexity.

The static analysis techniques can be applied to a wide variety of candidate software types. These types can be either developing software or existing legacy systems in maintenance, whether prototypes, incremental builds, or a full multi-phased lifecycle. These risk assessment techniques are applicable to all domains, from real-time process control to information systems.

Source code will be submitted to the DII Engineering Office as part of the candidate evaluation process. The complexity and quality metrics, contained in Table 2-2, will be collected from the source code using a COTS tool. These metrics are described in detail in Appendix C.

Table 2-2. Component Metrics Dictionary

Component Metric	Definition	Threshold Value
Halstead's Length	Measure of modularity of the design.	350
Halstead's Difficulty	Measure of the difficulty of developing the component.	50
Cyclomatic Number	Measures the number of testable paths in a component	15
Essential Complexity	Measure of the structure of the testable paths in a component.	7
Design Complexity	Measures the complexity of the control flow implemented by the design.	10
Source Lines of Code (SLOC)	The physical length of a component.	70
Control Density	Measures the percentages of control structures in a component.	.33

Table 2-2. Component Metrics Dictionary (cont.)

Component Metric	Definition	Threshold Value
Maximum Number of Levels	Measures the depth of IF..THEN..ELSE Nests in components.	7
Number of Branching Nodes	Measures the number of "gotos" or number of abnormal exits from control structures and loops.	7
Number of Input/Output Nodes	Measures the number of ways in and out of a component.	2

The values of these metrics are combined to provide information on the Maintainability, Correctness, and Reliability of the code. These three criteria comprise the Quality Criteria Profile, which is described in detail in Appendix C.

The set of metrics shown in Table 2-3 is collected to assess the complexity of the implementation of a specific call graph architecture. This set of metrics assesses the data and control flow complexity for the call graph in a manner similar to a component assessment.

Table 2-3. Call Graph Metrics Dictionary

Call Graph Metric	Definition	Threshold Value
Hierarchical Complexity	A measure of the average number of components on a level.	5
Structural Complexity	The average number of calls per component in the call graph.	3
Average Paths	The average number of paths per node in the call tree.	2
Number of Levels	The number of levels in a claa tree.	9
Entropy	A measure of orderliness in execution of the components in a call graph.	3

As an additional information product for each application, the metric analysis results and risk population information available from the QCP is detailed in a Complexity Analysis and Risk Assessment Report. This report provides a view of system quality at the component level and is

intended to support DII planning activities to provide recommendations and guidance for the developers.

Specifically, each COE candidate's Complexity Analysis and Risk Assessment report contains the following information:

- A description of the assessment process.
- A ranking of components in the high to medium risk categories.
- A statistical overview of each subdirectory (Actuarial Profile).
- Recommendations and guidance for ongoing software-related activities and process improvement.
- Assessment of future software-related difficulties and risks.
- A listing of the languages for each build with the number of components in each, the number of SLOC in those components, and the percentage of the total system contained in each.
- For each build, by language:
    - The percentage of components and corresponding statements that fall within acceptable risk limits for Maintainability, Correctness, and Reliability.
    - The percentage of components and corresponding statements that fall within acceptable limits for program length, number of statements, component structure and cyclomatic number, four key measures of component quality and complexity.
    - An indicator of the complexity of the call structure in the specific architecture.
    - The percentage of components and corresponding statements that belong to medium-to-high risk populations in the QCP.
    - The percentage of components and their associated statements exceeding the thresholds of key component quality metrics.
    - A measure of risk as determined by the number of components in common assigned to the high risk population interval for each of the three Criteria. This population is the Minimum Set (MS).
    - A listing of those components that are in the medium and high risk population intervals for all three Criteria in the QCP.

- A listing of those call roots with structures of such complexity that integration testing is likely to be incomplete and which present a risk to the system.

- A list of components exceeding the thresholds for key metrics.

2.4 Static Analysis Metrics Dictionary

The metrics collected for the COE software quality compliance assessment are listed in Table 2-4 and are widely accepted as indicators of software complexity and quality. Components with metric values exceeding the threshold values are viewed as suspect. The threshold values are determined by accepted industry values for good software engineering practices. Specific development organizations may use ranges of metric values. The values for the lower and upper limits of a range are determined by the organization's metric database of past projects.

The references contained in Appendix B are helpful in understanding the basic principles of metrics analysis techniques and the interpretation of threshold values. A review of the literature focusing on the fundamental metric definitions, collection, and analysis techniques is contained in *Software Engineering Metrics and Models* by Conte, Dunsmore, and Shen. This reference is critical to understanding specific metrics definitions and the necessity of a combined analysis of different types of complexity metrics.

Additional analysis of source code is required to understand the conditions causing the metric values to exceed the thresholds. Using a single metric value in this dictionary is not recommended or valuable. A combined analysis of the set of metric values in Table 2-2 is required to provide the information necessary for risk assessment.

2.5 Quality Criteria Profile

The formulas defining the three elements of the Quality Criteria Profile each use a specific set of metrics from the dictionaries. The metrics in each formula were selected based on the importance of the information for each element. The references in Appendix B are helpful in understanding the relationship of each metric to the formulas. The distinct advantage of these metrics is the invariance to the implementation language. For example, the cyclomatic number is the measure of the number of independent paths through a component regardless of the implementation language.

After the complexity metrics in the dictionary are collected from each COE candidate, a criterion is calculated by assigning a value of one to the value of each metric that falls within the acceptable range and zero if otherwise. This value is multiplied by the weight assigned to that metric in the formula. The total of the weighted values divided by the maximum possible value gives the percentage that determines the criterion risk category. The values of the percentages of the populations of components and corresponding SLOC are used in the DII Software Quality Compliance Checklist contained in Appendix A.

The QCP information quantitatively describes Maintainability, Correctness, and Reliability. This information, collected for each COE candidate, provides the DII Engineering Office with an automated, non-intrusive assessment of the level of Software Quality compliance.

The QCP and architectural complexity formulas and risk assignment numerical intervals are contained in Appendix C.

2.6 Risk Population Assessment Process

The information products provided by the QCP formulas are used to filter the components in a COE candidate into one of four risk intervals, as shown in Figure 2-1. The purpose of this filtering process is twofold:

- Determine if the percentage of the population, measured in both number of components and corresponding SLOC, is below the 20 percent rule of thumb.

- Pinpoint the components causing these populations to exceed the 20 percent rule of thumb.

For the purposes of the COE Software Quality Assessment process, the DII Engineering Office focuses on the components in either the medium or high risk interval. Typically, this combined percentage should be less than 20 percent of the total population of components or corresponding SLOC.

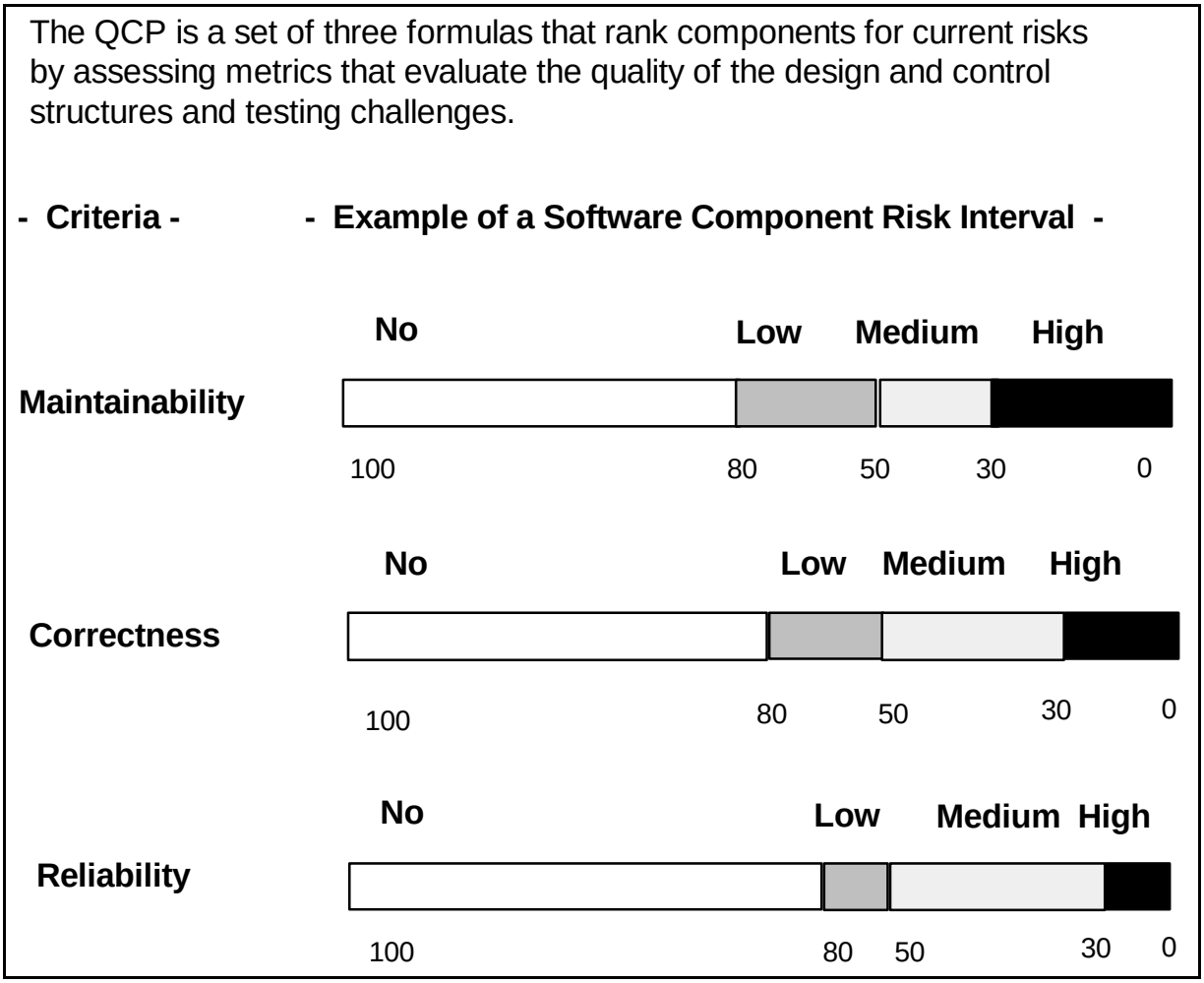

Figure 2-1. QCP Risk Population Intervals

The assessment process of the risk population information provided by the QCP uses three non-intrusive techniques, as shown in Figure 2-2, to further filter and assess the current medium-to-high risk components in each of the COE candidates.

This assessment process also projects the medium-to-high risk components resulting from future software-related activities (such as continued prototype development or maintenance) by the applications developers. The increase in the current medium-to-high risk populations occurs due to both insertion of new complex components and modifications of complex components in the application's baseline.

The three COE candidate assessment processes, shown in Figure 2-2, are:

- Medium-to-high Risk and Minimum Set (MS) Population
- Emerging Risks
- Complexity and Quality Factors.

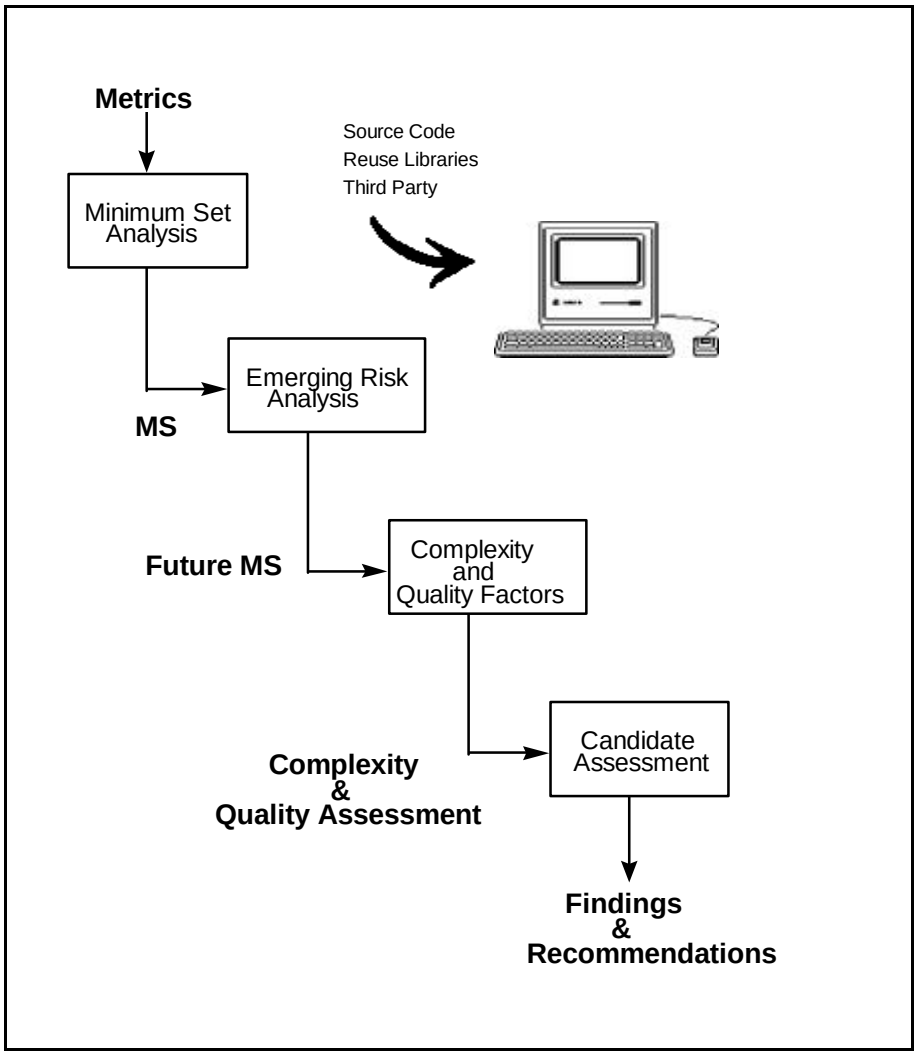

Figure 2-2. Steps in the Static Analysis Process

2.6.1 Minimum Set Analysis

The first technique in the static analysis process is the identification of the components in the Minimum Set. This technique, shown in Figure 2-3, identifies the common components in only the high risk populations for each of the three Criteria in the QCP. The components in this population are considered to represent the highest risk to all applications software-related activities, such as integration and continued modifications by the developer.

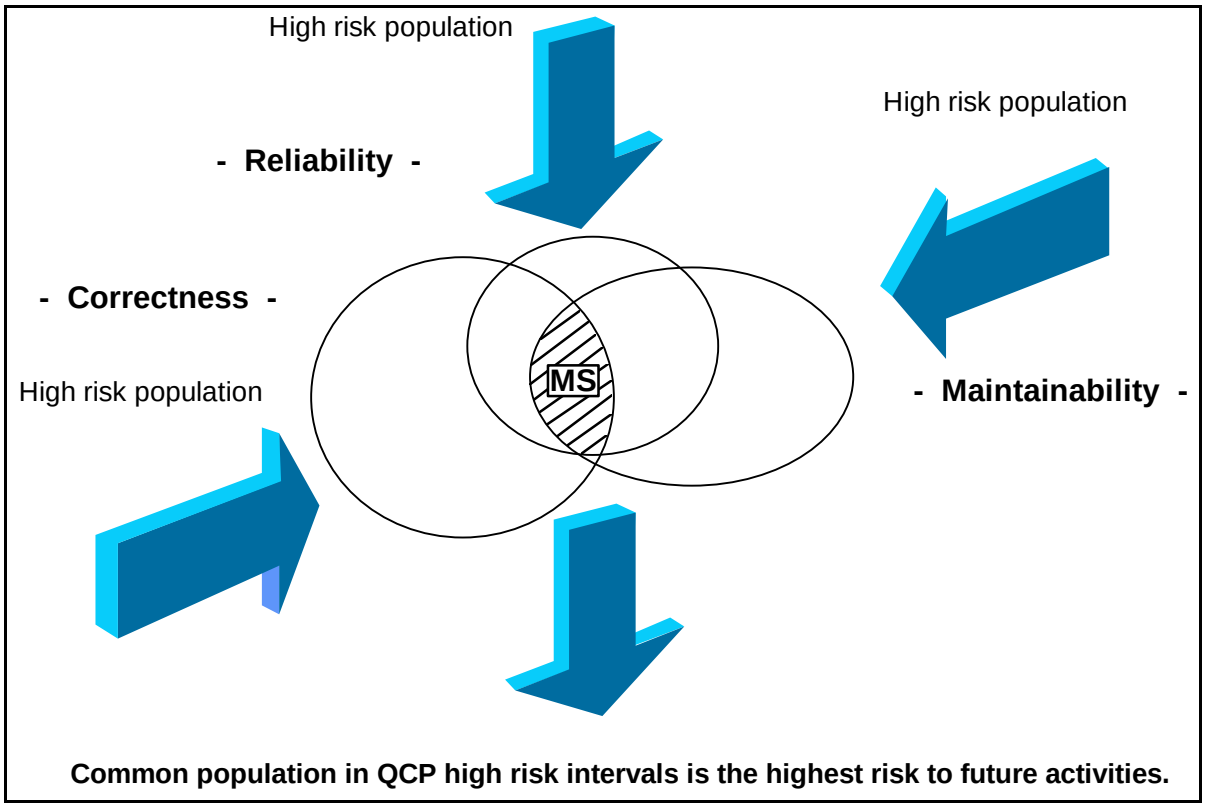

Figure 2-3. Minimum Set Analysis Process

Typically, the percentage of components in the MS population is low. For DII integration planning purposes, risk values of Low (L), Medium (M), and High (H) are assigned to each CSC/CSCI by first the computing the percentage of components and corresponding SLOC in the MS for the candidate.

The value of the population percentage, measured for either the percentage of components or the SLOC in those components, is preliminary assigned either L, M, or H by the rules shown in Table 2-4:

Table 2-4. Rules for Assigning Risk Values

Risk	Population Percentage
L	MS%<4%
M	4<=%MS<6
H	6<=%MS

Inspection of the data in Figure 2-4 depicts a COE candidate with nine CSCs. The nine CSCs are shown along the horizontal axis. The information in this figure indicates only two of the nine CSCs, CSC_3, and CSC_6, do not have any high risk interval populations, which are shown on the vertical axis. Seven of the nine CSCs have components in the high risk interval for at least one of the three Criteria, shown as different patterns in the bars. Currently only CSC_4, CSC_5,

and CSC _9 have MS populations. The components in the MS population represent the highest risk to integration activities, as determined by this assessment process. Further inspection of this figure indicates CSC_8 has high risk population in all three QCP Criteria, but currently the MS population is zero. This information alerts the DII Engineering Office of an existing pocket of complexity that will grow with continued software modifications.

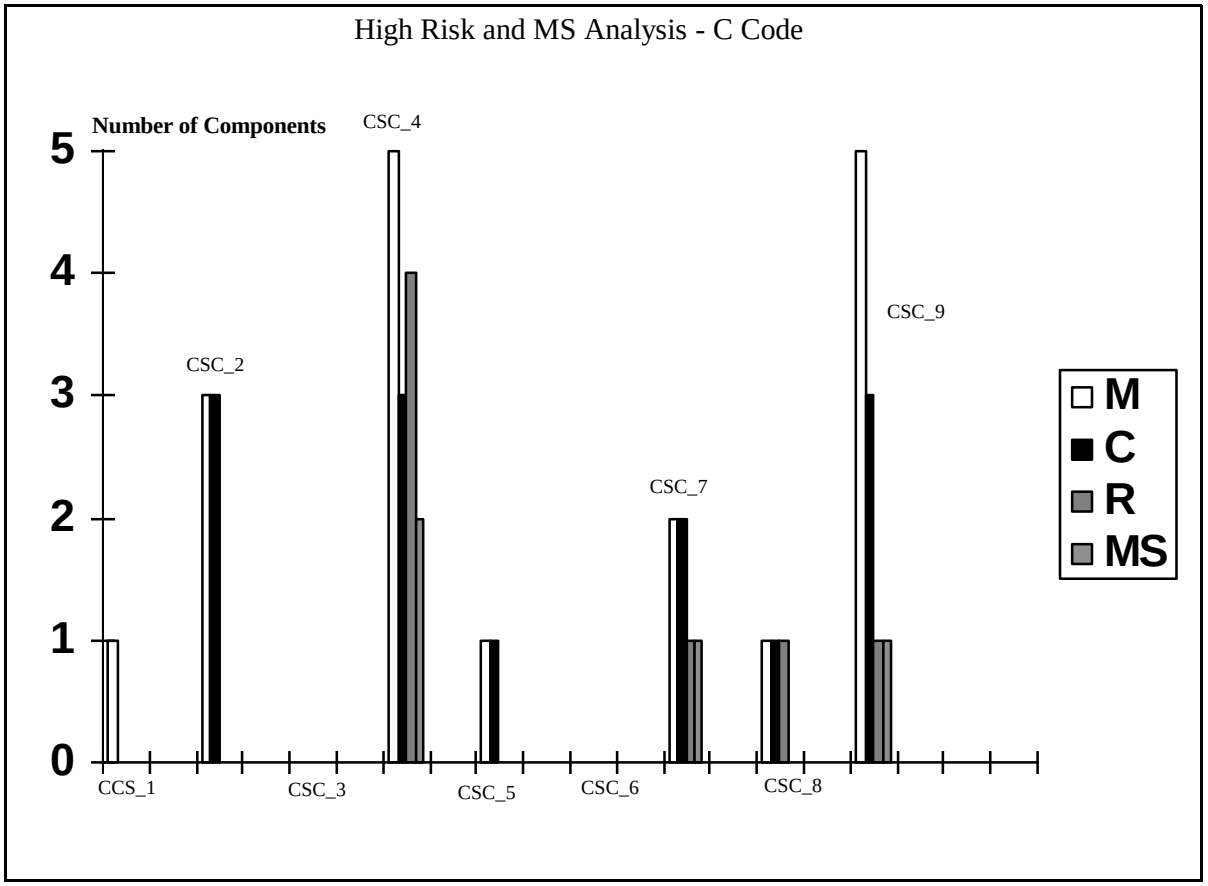

Figure 2-4. Example of a High Risk Population and Minimum Set Distribution

Figure 2-5 contains an example of information on the distribution percentage of SLOC in a MS population. In this example, each component in the MS population is identified, and the percentage of SLOC in the component is also given. Inspection of CSC _4 indicates the presence of two very large components, labeled 1 and 2. These two components should be considered for re-engineering to mitigate the risk. The specific re-engineering activities should reduce the size, loss of modularity, and complex control structures.

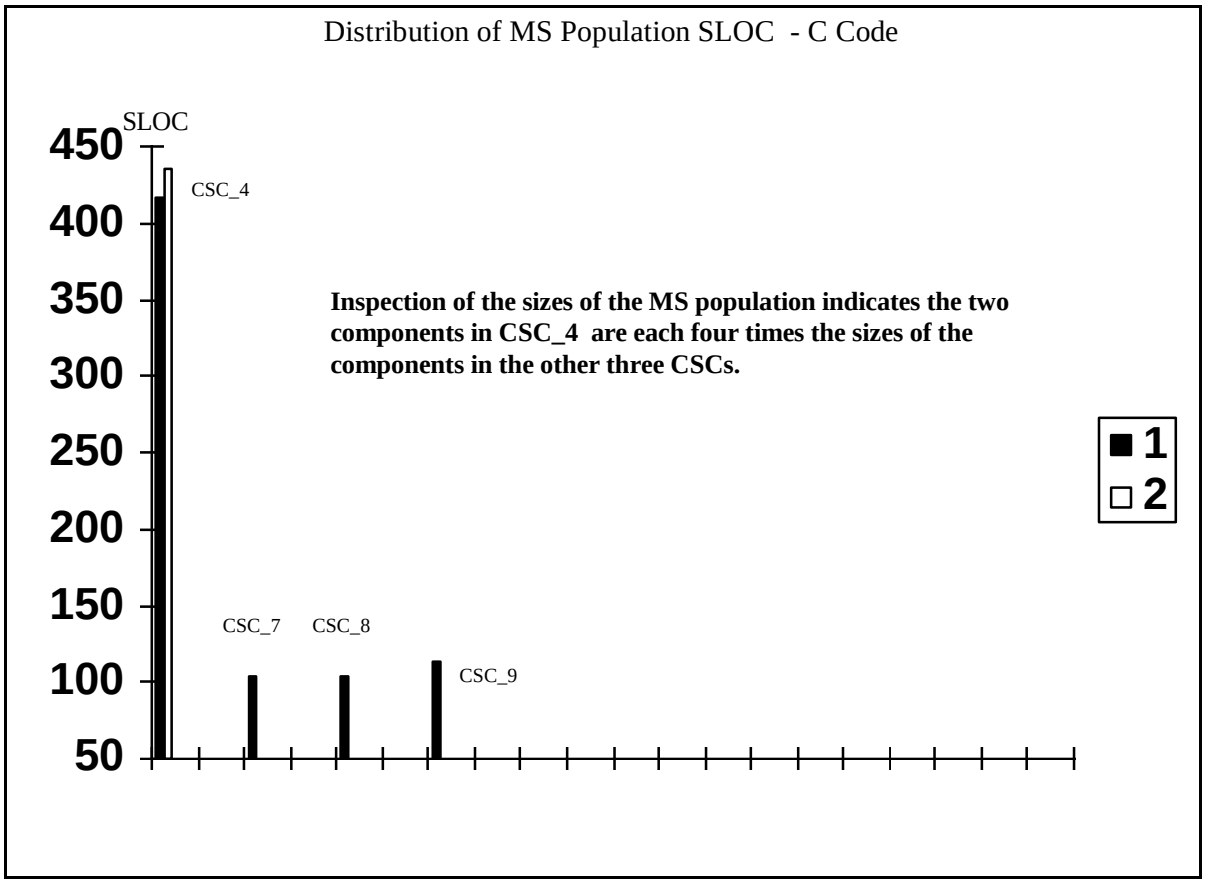

Figure 2-5. Example of a Minimum Set Population

2.6.2 Medium-to-High Risk Population Analysis

The medium-to-high risk population assessment provides more detailed information on the current population as measured in both percentages of number of components and corresponding source lines of code. As always with static metrics analysis, the values of both number of components and corresponding SLOC are important. This pair of metrics is extremely valuable in understanding the current proportion of the software in a COE candidate actually in the medium-to-high risk interval. For example, using both of these values helps to indicate if the existing problems are a few, large components; many small, complex components; or a combination of large and small components.

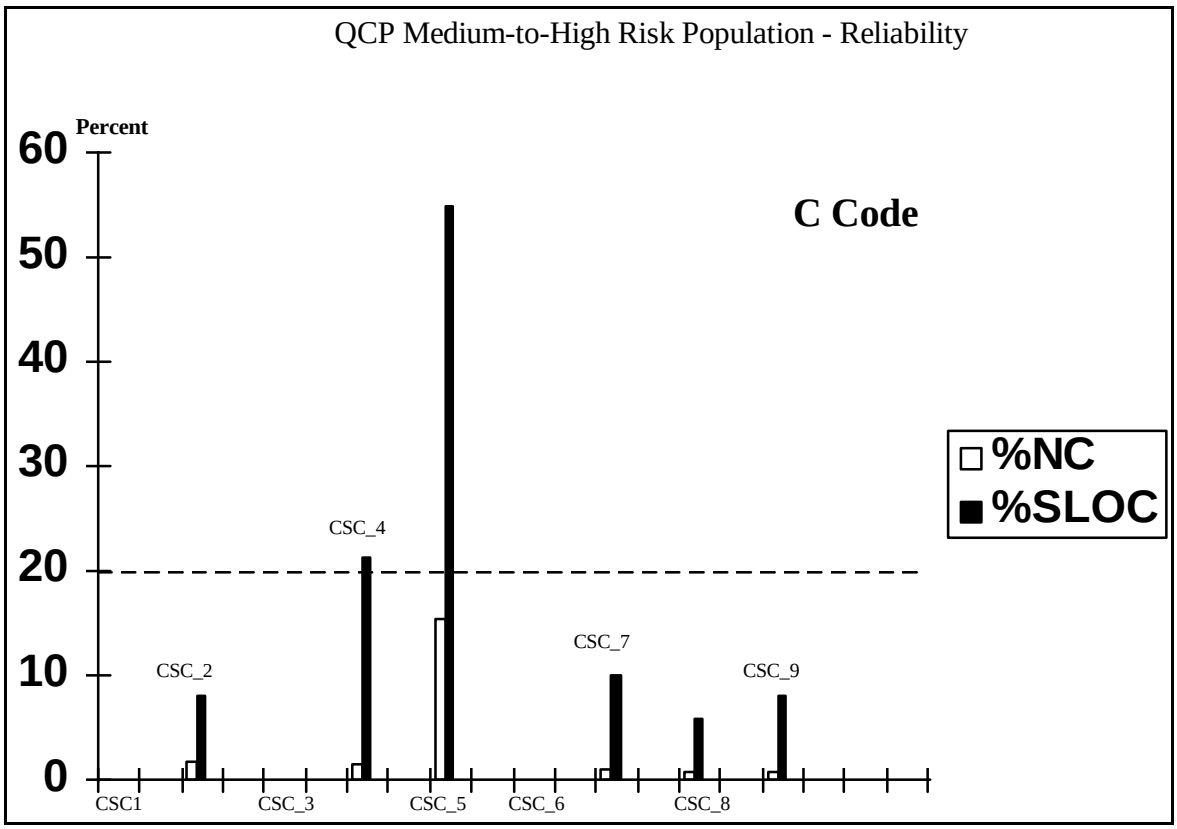

Figure 2-6. Medium-to-High Risk Population Percentage Distribution

Figure 2-6 provides an example of the medium-to-high risk population distribution for the Reliability Criterion of the QCP. In this example, only CSC_4 and CSC_5 have percentages of number of components or corresponding percentages of SLOC exceeding the 20 percent rule of thumb.

This information provides important inputs for the DII Engineering Office to prepare and evaluate risk reduction options for a COE application. Typically, complexity in software continues to grow as repeated modifications build upon existing complexity. As this medium-to-high risk population grows over time, the entire CSCI/CSC will gradually become a candidate for re-engineering or become unusable and unaffordable.

Figure 2-7 contains an example of a medium-to-high risk population analysis for a COE candidate. The combinations of medium-to-high risk intervals are listed starting with the highest (H-H-H), the Minimum Set, to lowest. In Figure 2-7, the percentage of components and corresponding percentage of SLOC in the CSCI/CSC are shown. Notice the large population currently in Redesign Redesign Reimplement (H-H-M) intervals.

Future software modifications of the code in these components in the H-H-M risk interval will cause migration to the H-H-H risk interval unless renovation reduces the complexity. The result of the migration of components will result in over 20 percent of the software being in the medium-to-high risk population.

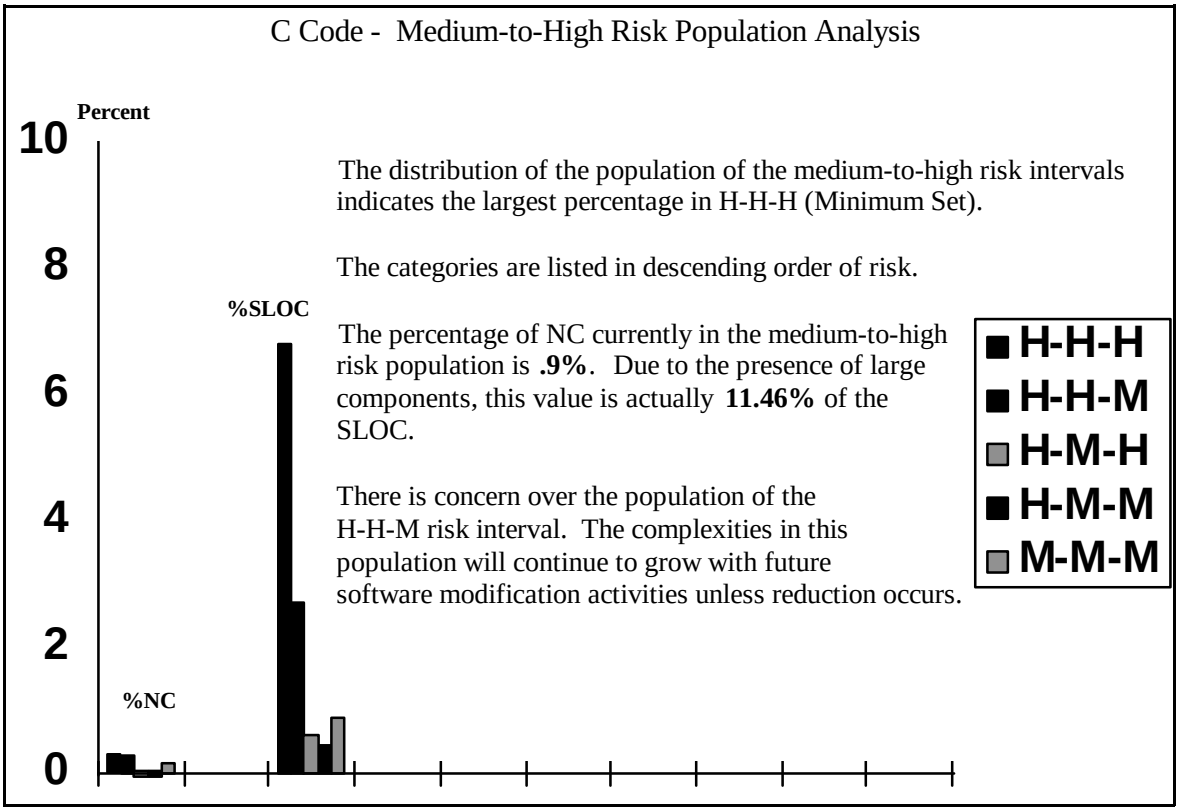

Figure 2-7. Example of a Medium-to-High Risk Population Analysis

After developing the baseline for each candidate, the DII Engineering Office tracks increases and reductions in risk by a change analysis of the medium-to-high risk population. Specifically, not only are increases or decreases in percentages tracked, but a correlation to changes in number of components and SLOC are necessary to understand the impact on the risk. This information is useful in assessing the stability and maturity of the application.

2.7 Emerging Risk Analysis

The second analysis technique focuses on obtaining an estimate on the potential increase in the QCP high risk populations if future software modifications to candidates occur without renovation of the current complex control and design problems. This emerging risk analysis, depicted in Figure 2-8, uses the differences between the current high risk populations and the current combined low and medium risks populations.

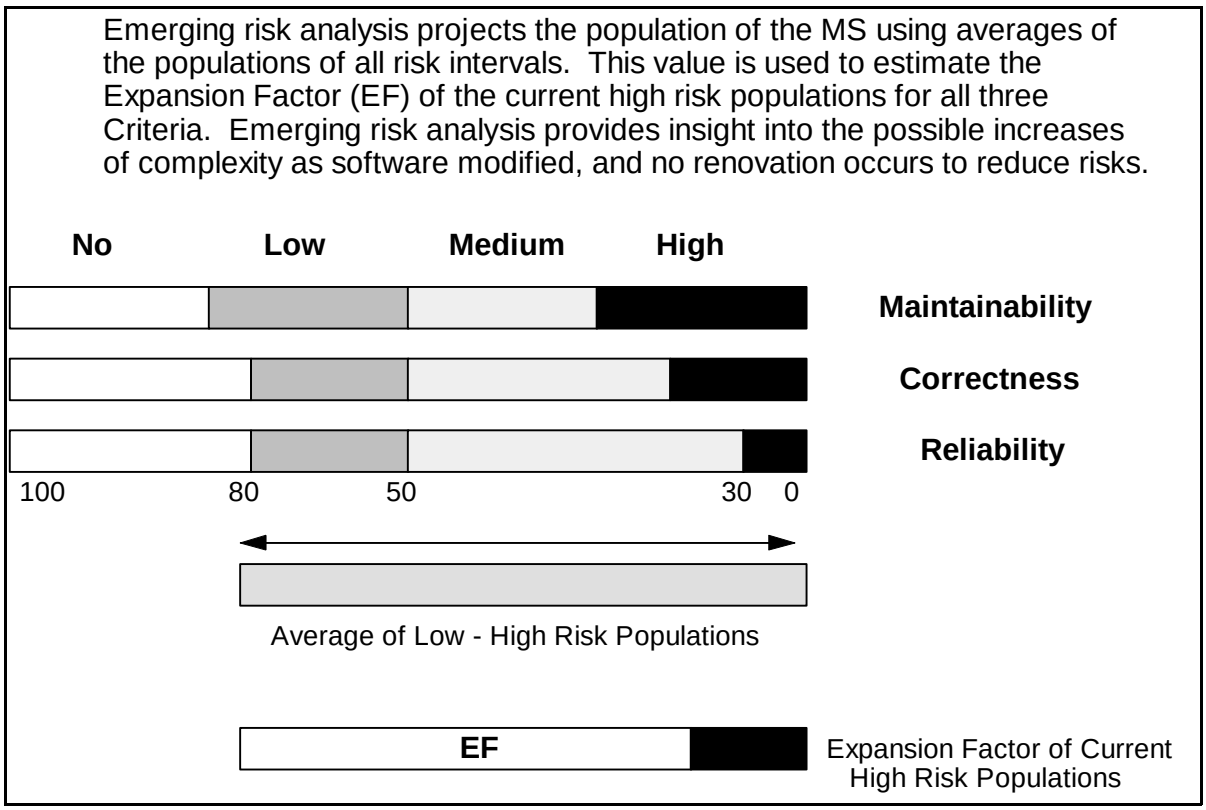

Figure 2-8. DII Software Emerging Risk Analysis

The Emerging Risk and Expansion Factor analysis represents the possible "avalanche" of increasing risk if code modifications are performed on applications without prior renovation based upon the risk population analysis.

Figure 2-9 provides an example of an Expansion Factor Analysis for the nine CSCs in the COE candidate. For each CSC, the value of the Expansion Factor and size of the CSC are shown. Notice the size of the CSC does not seem related on the EF value. CSC_1 and CSC_2 seem to contain higher values of EF for the corresponding physical sizes.

The Emerging Risk estimation technique, described in Appendix C, does not assume the addition of new, complex components into the baseline nor the deletion of components from the baseline. The technique assumes equal likelihood of all components undergoing modifications and is intended to provide a worst case outlook.

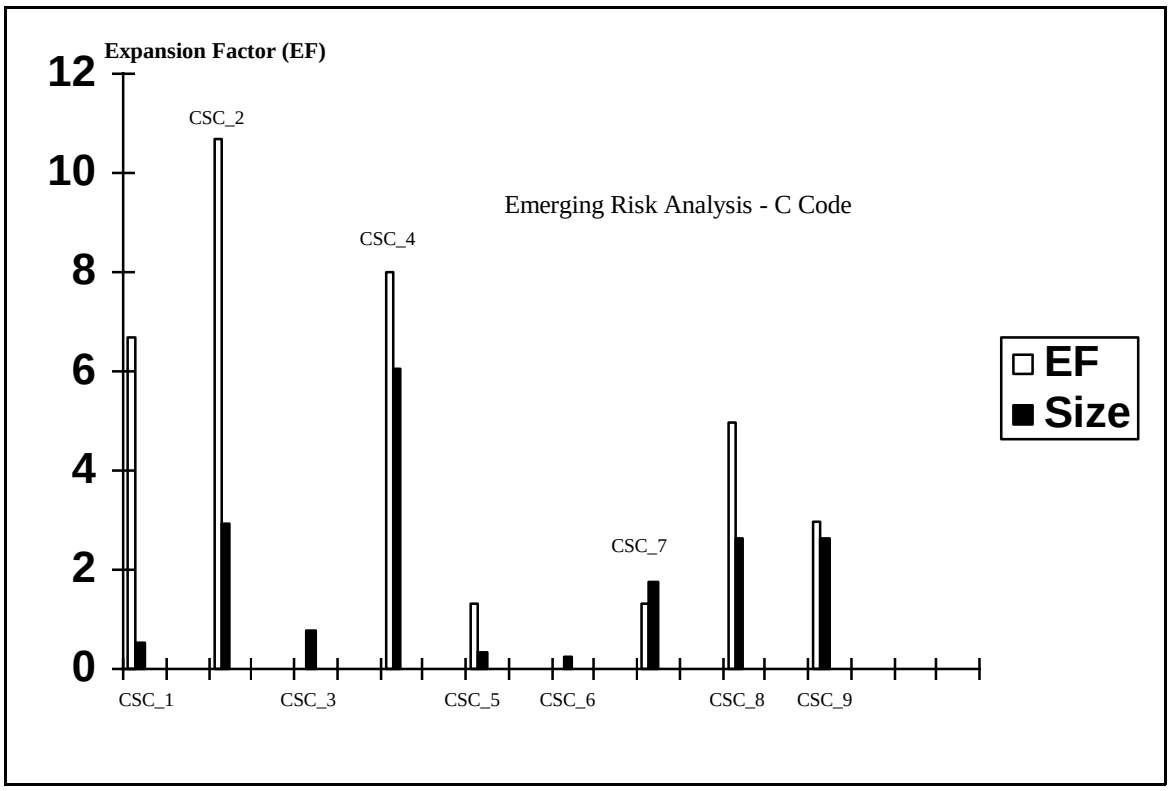

Figure 2-9. Example of an Expansion Factor Analysis

2.8 Complexity and Quality Factors Analysis

The third static analysis technique provides information on the complexity and quality factors of the components in an application. This technique, depicted in Figure 2-10, obtains estimates of the Control, Design, Size, and Process factors by averaging the combined populations for the low to high risk intervals for a set of key metrics.

These key metrics were selected to provide information on the current populations exceeding thresholds in design modularity, physical size in SLOC, control complexity, and the use of unstructured control structures. High numbers of applications components exceeding the thresholds for each of these four metrics can alert the DII Engineering Office in planning future integration schedules.

Specifically, as shown in Figure 2-10, these factors are obtained by averaging the total risk populations for the cyclomatic number, Program Length, SLOC and essential complexity metrics. Further analysis proceeds by evaluating the relationship of design versus process, and design versus size and cyclomatic number to detect the most suspect areas.

Four key metrics provide an analysis of different, but related complexity and quality factors. The analysis process uses the average of the total populations in the low to high risk intervals for each of the four key metrics. This analysis provides management insight into the four key complexity and quality problems currently plaguing the software and the type and cost of renovation needed to reduce the risk.

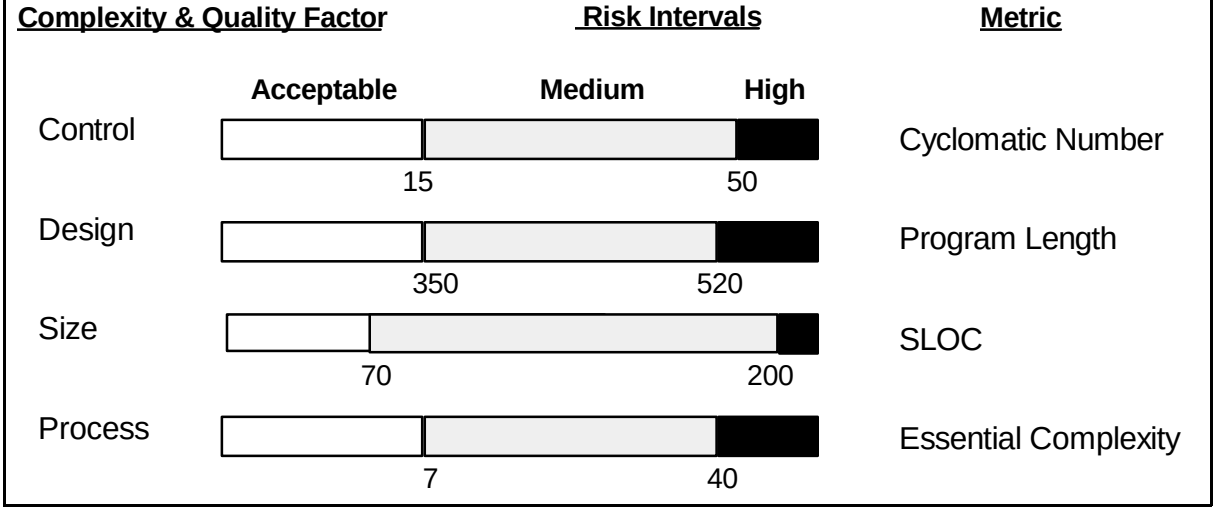

Figure 2-10. DII Software Complexity and Quality Analysis

Figure 2-11 contains an example of the results of a complexity and quality factors analysis for a COE candidate. For two CSCs, A and B in the candidate, there are two values for each of the four complexity and quality factors. The first value is the percentage of components exceeding the threshold, and the second value is the corresponding SLOC associated with the percentage of components.

The bar graphs in Figure 2-11 indicate CSC_1, CSC_2, CSC_4, CSC_5, and CSC_9 have exceeded the 20 percent rule of thumb. The results of this information alerts the DII Engineering Office to the testing challenges associated with this candidate, as well as the increased likelihood of embedded faults.

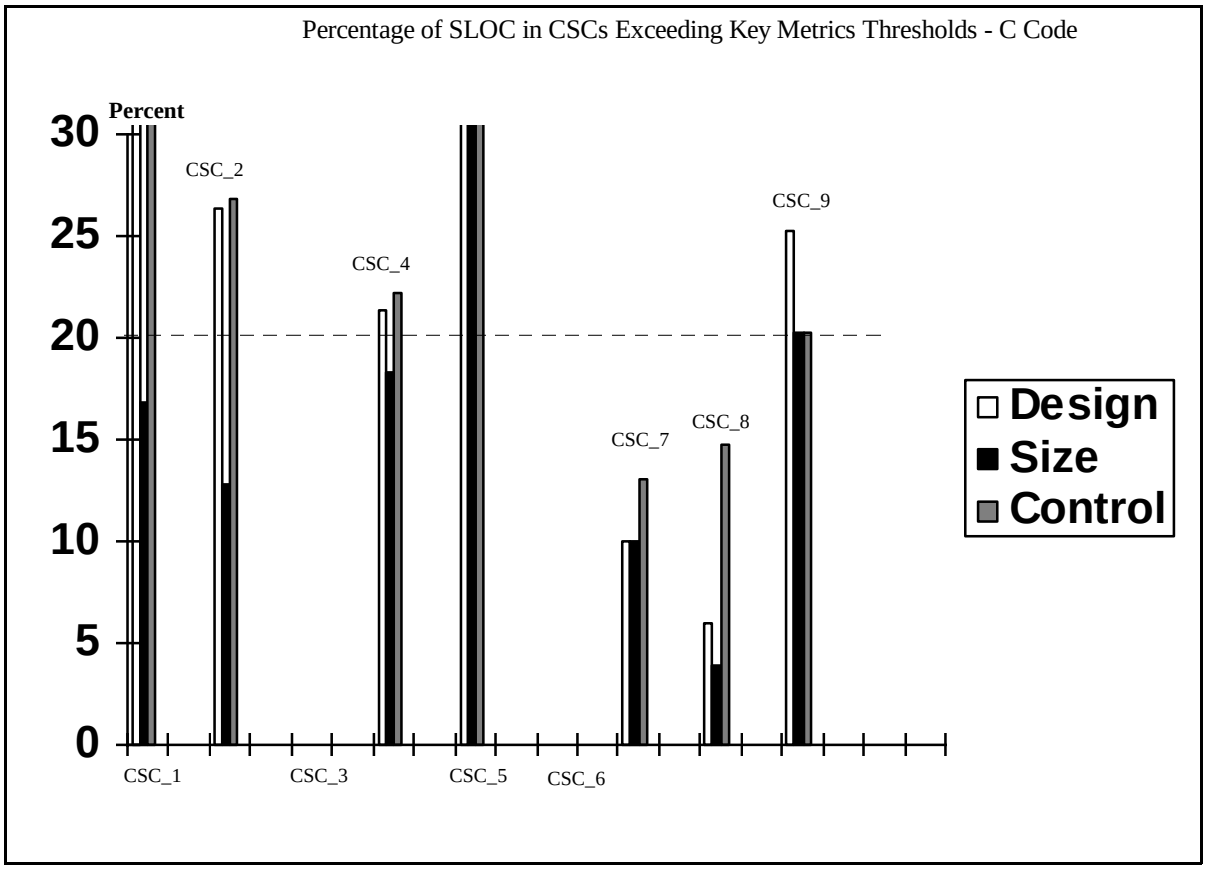

Figure 2-11. Example of Software Complexity and Quality Analysis Information

2.9 Static Analysis of API Code

The DII COE I&RTS provides guidance on public APIs for use during the building of applications for the COE. One of the requirements for achieving Level 7 compliance is the degree of usage of public APIs. To function as a measure of progress towards meeting this goal for both DII and developers, the source code libraries and MAKEFILES will be assessed to locate references to non-public API libraries. Frequent use of non-public APIs will be assessed and used as information to determine the progress and schedule of meeting the target level of COE Software Quality Compliance.

2.10 Trend Analysis of Application Versions Metric and QCP Risk Populations

The trend analysis of metrics and risk population databases for COE candidates provides information on the changes in the complexity of the software. These changes result from modifications to existing software and/or introduction of new components. During DII integration planning activities, significant increases in medium-to-high risk populations are important factors in planning resources required to field new versions.

Detection of increases of components in medium-to-high risk populations also may signal problems in planned integration testing activities. Specifically, integration test schedules may not be adequate for applications with significant increases in the medium-to-high risk populations. Typically, significant increases in the populations of medium-to-high risks result in an increase in the number of SPRs.

The trend analysis process evaluates the following information:

1. Population growth in number of components and corresponding SLOC.

2. Increases or decreases in the population of the combined medium-to-high risk intervals for the number of components and corresponding SLOC.

3. Increases or decreases in the percentages of components exceeding the thresholds for key metrics.

4. Increases or decreases in reported application SPRs.

5. Increases or decreases in the percentages of components and SLOC containing OS-specific or implementation-specific language statements.

Figure 2-12 contains a sample of a trend analysis of the growth of the medium-to-high risk population of a application. The steady increase of the value of the percentages of both the number of components and corresponding SLOC indicates an increased integration risk in the COE functions.

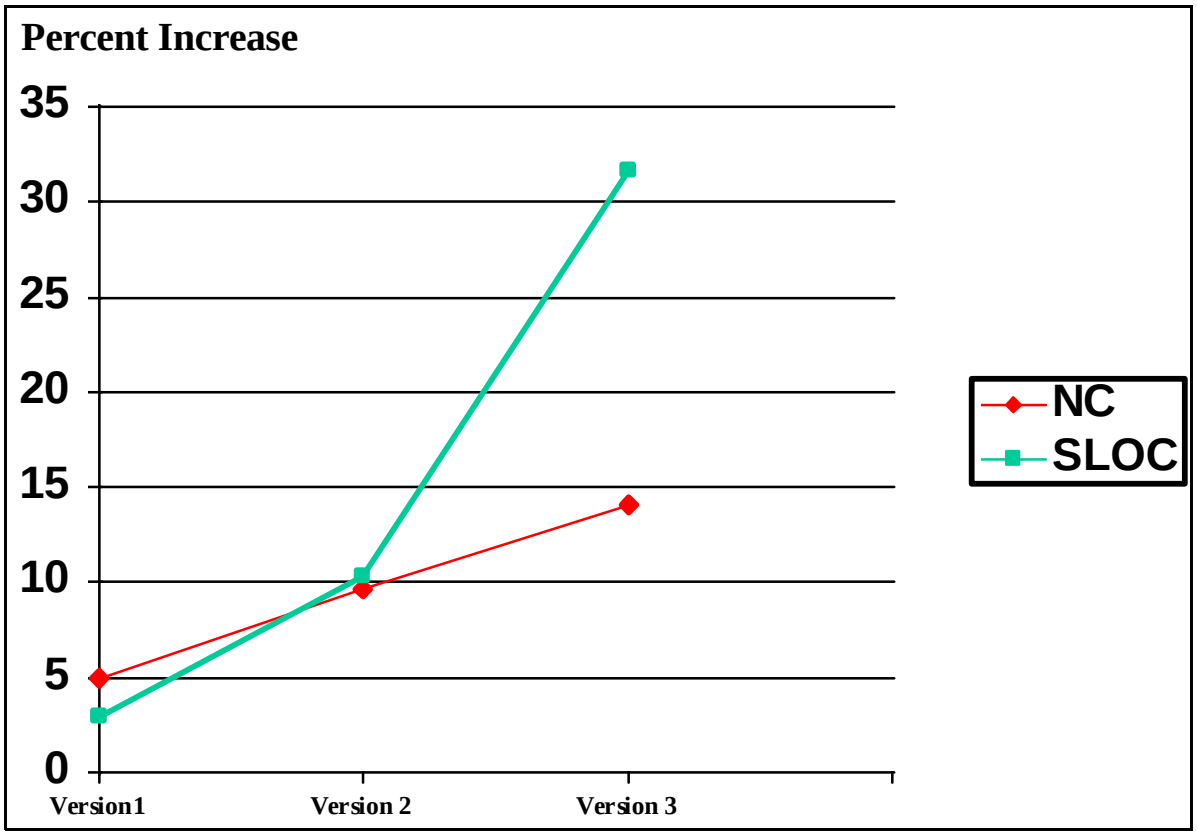

Figure 2-12. Example of Trend Analysis of Medium-to-High Risk Population Growth

Figure 2-13 contains an example of a trend analysis of a complexity and quality analysis for a function in the COE. In this figure, the percentages of the number of components (NC) and corresponding SLOC exceeding the 20 percent rule of thumb are shown for two versions of a candidate, V.1 and V.2, for each of the four complexity and quality factors. Notice the growth of the sizes of the components, number of control paths, and usage of unstructured logic. Continued growth of these population of components exceeding the thresholds of these key metrics will increase the risk.

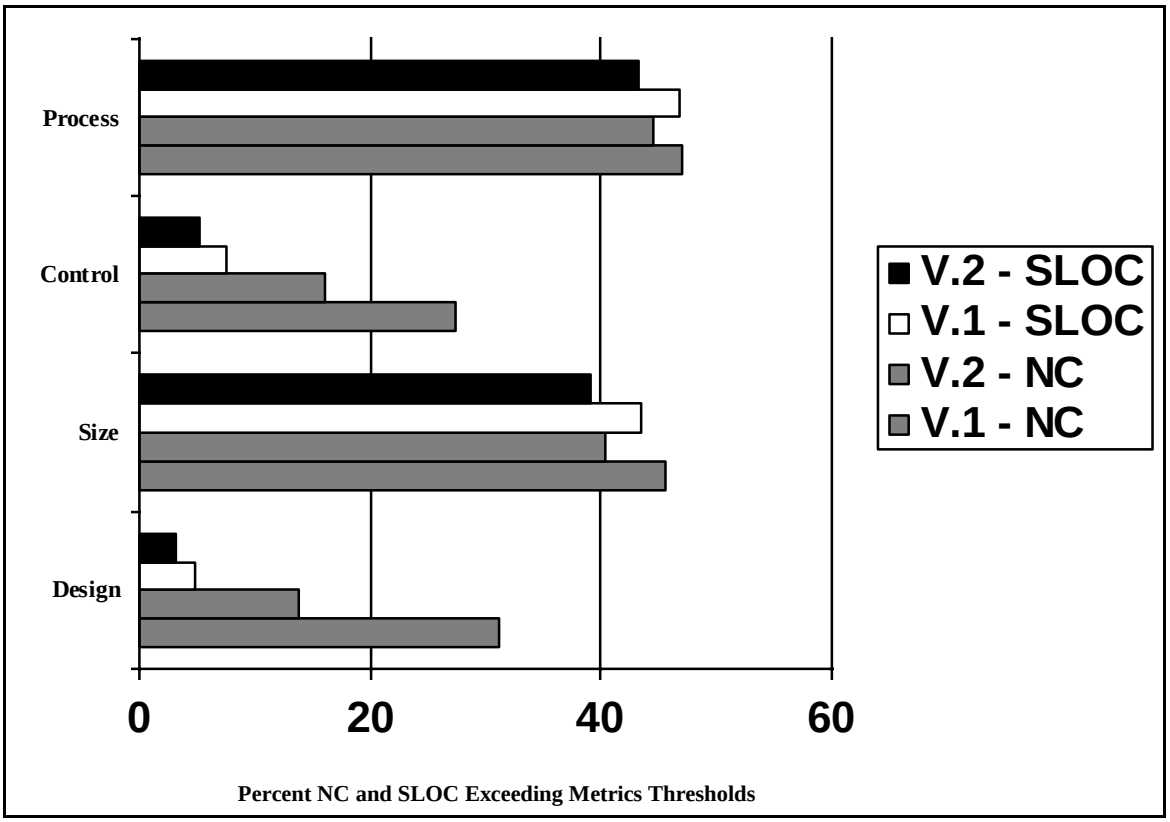

Figure 2-13. Example of Trend Analysis of Software Complexity and Quality Analysis

2.11 Integration Test Effectiveness Analysis

The metrics collected from the complexity, quality, and portability analyses and conformance to compliant APIs provide information obtainable from static analysis of the software. Extending this set of metrics to include information obtainable from a dynamic analysis provides a more encompassing software compliance assessment process.

The approach to obtaining dynamic information without actually executing the code is to use the data obtainable from the SPRs generated during integration testing and field performance. This information provides insight into the stability of the code delivered to the DII Engineering Office. Examples of useful metrics obtainable from SPRs are the number of problems; potential operational impact (Type 1, 2, 3, and 4); and, if possible, the amount of code changed to repair the problems.

The risk information obtained from the static analysis of the current version of application source code is synthesized with the information obtained from the dynamic analysis as well as information from trend analysis of QCP medium-to-high risk populations of the previous versions. This synthesis provides insight and guidance in assisting the DII Engineering Office and application owners in further assessing integration readiness.

The further insight into the potential risk in the execution of a CSC/CSCI is provided by inspecting the change of the percent of components in the MS. This value and the names of both the components in the previous version and the new components represents a high degree of vulnerability identified to the COE. Specifically, the components in the MS most likely contain residual faults that will be difficult and time consuming for the application owner to repair upon occurrence either during the integration or detection by the user. The synthesis with the information provided by the dynamic metrics will pinpoint those CSCs/CSCIs with high frequencies of repairs and occurrences of Type 1 and 2 problems.

Although the components in the MS represent the greatest risk to COE, the components in the populations of the other combinations of medium and high risk intervals for the QCP (described in Subsection 2.6.2) also require special emphasis in the integration testing planning and execution process. The integration testing planning process will also give special emphasis to components in the medium-to-high risk populations, and the test results will be evaluated to assess the impact of software changes.

As part of the application source code submission process, test cases supplied by the application owner will contain information on the QCP risk intervals of the components in CSC/CSCI paths executed for each test case as well as other dynamic metrics. This information will be used as a predictive, a trouble-shooting risk mitigation technique, and level assessment input.

2.11.1 Application Portability Analysis

The evaluation of applications to assess COE software quality compliance is based upon an assessment of four factors that heavily influence portability. These four factors are:

- Use of non-compliant tools and COTS
- Use of non-compliant APIs
- Percent of OS-specific extensions to the implementation language
- Percent of non-POSIX functions used.

Using a COTS tool to perform static analysis of the source code detects OS-specific subsets of language extensions used in the application. Unless usage of language extensions is managed and minimized by the application owner's coding standards and conventions, the components may contain large occurrences of machine-dependent code. Typically, the extensions provide a specific functionality that, if not available on a new platform, will require additional code generation for porting. The more language extensions are used, the higher the difficulty and costs of porting to other platforms. Additional risk is added to the GCCS integration process if code modifications necessary for porting adversely impacts the application's medium-to-high risk populations.

The analysis process of application portability includes collection and analysis of machine/language dependencies metrics. These metrics will be collected as part of the complexity and quality metrics process. Specifically, language extensions provided by specific compilers and operating system environments will be counted and expressed as percentages of number of

components and SLOC. The analysis will also correlate these percentages to the medium and high risk populations and components. The DII Engineering Office will use this information to mitigate risks in planning schedules to transition to new platform technologies.

This page intentionally left blank.

3. COE SOFTWARE QUALITY EVALUATION SCHEDULE

The COE integration schedule is shown in Figure 3-1 and contains the evaluation activities for the candidates.

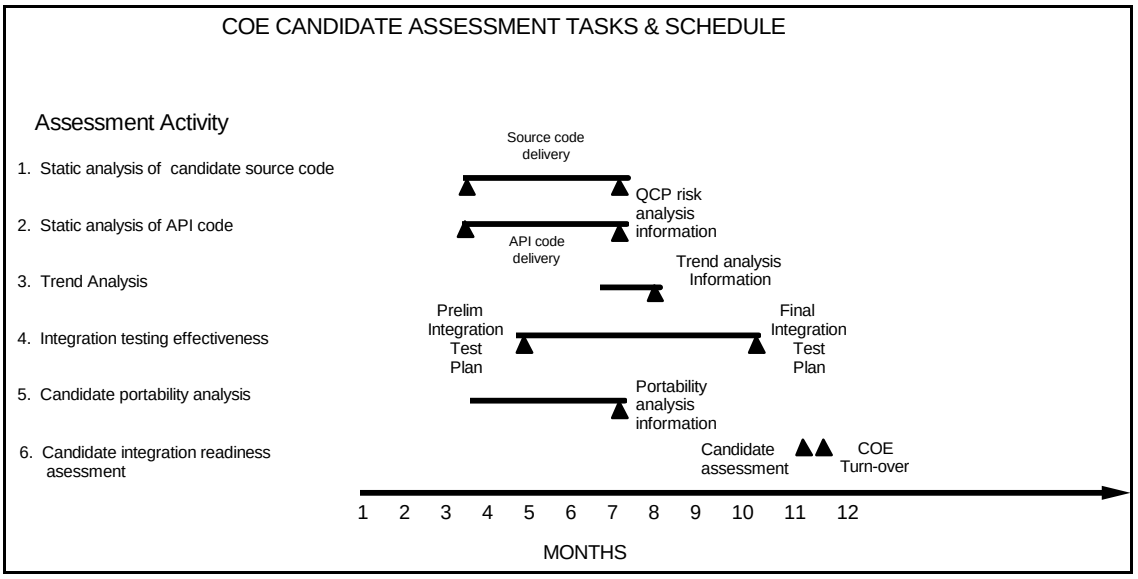

Figure 3-1. COE Candidate Evaluation Schedule

Each COE developer, as specified by DISA, will provide source code and API code for static analysis. As part of the integration risk management process, these applications will be analyzed to build the baseline of QCP risk populations and assess portability of the source code and the complexity of the API code. In subsequent deliveries of application version updates, a trend analysis will provide information on changes in the previous populations of medium and high risk components and portability.

The results of the baseline and if applicable, the trend analysis of the QCP medium-to-high risk populations will be delivered to the COE integrator for possible revisions to the test plan, as shown in Figure 3-1. Upon completion of the COE integration testing activities, the DII Engineering Office will perform an analysis of the SPRs and the QCP medium-to-high risk populations to determine the effectiveness of the testing activities.

Prior to COE turn-over to the Mission Applications developers, the DII Engineering Office will prepare a COE Integration Readiness Assessment Report. This report will contain an assessment of the risks of the COE software and identify, by component name, medium-to-high risk populations.

This page intentionally left blank.

APPENDIX A: DII SOFTWARE QUALITY COMPLIANCE CHECKLISTS

Software quality is the fourth category in the four DII COE compliance categories. The assessment process in this fourth category supports the seamless segment integration goals by:

- Ensuring usage of compliant APIs, COTS, and POSIX.
- Minimizing vulnerability to the ever-present and unintentional introduction of residual faults in third party software.
- Easing the burden of source-level integration.

The information gained from this empirical and repeatable assessment process provides benefits to both the candidate owners and DISA by:

- Determining the degree of portability currently in the software
- Identifying degrees of integration risk for CSCIs and pinpointing individual suspect components for renovation.

The compliance evaluation process consists of both automatic and manual activities. Specifically, a COTS tool provides the information necessary for Levels 1 - 4 from a static analysis of the source code. Level 5 is evaluated using information supplied by the developer when the source code for the COE candidate is delivered.

This appendix contains a series of checklists organized by compliance level for Category 4. The levels range from compliance assessment with approved DISA APIs and COTS to risk interval population analyses, trend analysis, and test effectiveness. Each item shall be answered as *True*, *False*, or in the case of Level 5, *Not Applicable* (*N/A*).

The levels are listed in order of the degree of risk minimization associated with each level. For example, an application could use compliant APIs and COTS while the CSCIs contain excessive populations in the medium-to-high risk intervals. An application achieves the highest level of compliance by:

- Implementing only compliant APIs and COTS.
- Containing populations of components in the medium-to-high risk interval that are less than 20 percent of the total population of components and corresponding SLOC for each CSCI in the candidate.

- Containing populations of components that are less than the thresholds for key metrics for more than 80 percent of the components and corresponding SLOC in a CSCI in the candidate.

- Demonstrating effective testing of components in the medium-to-high risk interval.

- Minimizing populations of Type 1 and Type 2 software problems below the threshold.

- Reducing error occurrence rates to less than 7/KSLOC for each CSC in all of the CSCIs for each candidate.

If any of these six requirements are not met by an application, the owner of the candidate has the option to submit a plan to DISA for approval detailing specific activities and associated milestones to remove the deficiencies.

An application obtaining a Level 5 rating presents the lowest risk in Category 4 to achieving DII integration goals. The level of software quality compliance is determined by the highest numbered level for which there are no False replies.

An application achieving Level 5 will be accepted and advertised as a DISA-approved DII COE product. At DISA's discretion, applications that are Level 3 or Level 4 compliant may be accepted as prototypes and fielded at selected sites for evaluation purposes.

A.1 Conformance Analysis of API Code (Level 1)

T	F	1.	The number of CSCs in the CSCI using a compliant API is equal to 100 percent of the total population of CSCs for each CSCI.
T	F	2.	There is a DISA-approved migration plan number to re-engineer the source code to implement compliant APIs for all non-compliant CSCs for each CSCI.

A.2 Application Portability Analysis (Level 2)

T	F	1.	The number of CSCs in the CSCI using compliant tools and COTS is equal to 100 percent of the total population.
T	F	2.	The number of CSCs in the CSCI using OS-specific extensions is less than or equal to 20 percent of the total population.
T	F	3.	The number of CSCs in the CSCI using non-POSIX functions is less than 20 percent of the total population.

T F 4. There is a DISA-approved migration plan number to re-engineer the source code to implement compliant COTS for all non-compliant CSCs for each CSCI.

A.3 Static Metrics Analysis (Level 3)

QCP Risk Interval Populations

T F 1. The number of components in the combined medium and high risk population is less than 20 percent of the total population for each CSC.

T F 2. The SLOC in the combined medium and high risk population is less than or equal to 20 percent of the total SLOC for each CSC.

T F 3. The number of components in the Minimum Set population is less than 4 percent of the total population for each CSC.

T F 4. The Expansion Factor is less than two for all of the CSCs in a CSCI.

T F 5. There is a DISA-approved migration plan number to re-engineer the source code to reduce the medium-to-high risk population below the 20 percent thresholds for each CSC.

Analysis Complexity and Quality

T F 1. The number of components below the thresholds for 4 of the complexity and quality factors is less than 20 percent of the total population of components or corresponding SLOC in the CSC.

T F 2. The number of components below the thresholds for 3 of the complexity and quality factors is less than 20 percent of the total population of components or corresponding SLOC in the CSC.

T F 3. The number of components exceeding the thresholds for 2 of the complexity and quality factors is less than or equal to 20 percent of the total population of components or corresponding SLOC in the CSC.

T F 4. The number of components exceeding the thresholds for 1 of the complexity and quality factors is less than or equal to 20 percent of the total population of components or corresponding SLOC.

A.4 Trend Analysis of Complexity and Quality Factors and QCP Risk Populations (Level 4)

T	F	1.	The number of components below the thresholds for 4 of the complexity and quality factors remains less than 20 percent of the total population of components or corresponding SLOC in each CSC.
T	F	2.	The number of components below the thresholds for 3 of the complexity and quality factors remains less than or equal to 20 percent of the total population of components or corresponding SLOC in each CSC.
T	F	3.	The number of components exceeding the thresholds for 2 of the complexity and quality factors remains less than 20 percent of the total population of components or corresponding SLOC in each CSC.
T	F	4.	The number of components exceeding the thresholds for 1 of the complexity and quality factors remains less than 20 percent of the total population of components or corresponding SLOC in each CSC.
T	F	5.	The number of components in the combined medium and high risk population remains less than 20 percent of the total population for each CSC.
T	F	6.	The SLOC in the combined medium and high risk population remains less than 20 percent of the total SLOC for each CSC.
T	F	7.	The number of components in the Minimum Set population remains less than 4 percent of the total population of components for each CSC.
T	F	8.	The Expansion Factor is still less than two for all of the CSCs in a CSCI.

A.5 Integration Test Effectiveness Analysis (Level 5)

T	F	N/A	1.	The percent of Type 1 SPRs is less than 1 percent of the total number of SPRs for all the CSCs in a CSCI.
T	F	N/A	2.	The percent of Type 2 SPRs is less than 3 percent of the total number of SPRs for all the CSCs in a CSCI.
T	F	N/A	3.	Each CSC with greater than 20 percent of components or corresponding SLOC in the medium-to-high risk intervals has been adequately tested by the owner s SQA program.
T	F	N/A	4.	The software error occurrence rate per 1K SLOC is less than 7 for all CSCs in the CSCIs.

T	F	N/A	5.	There is a DISA-approved plan with risk mitigation information for each CSCI exceeding thresholds for Type 1, Type 2 SPRs, adequacy of testing, and error occurrence rate per 1K SLOC.

This page intentionally left blank.

APPENDIX B: BIBLIOGRAPHY

The references in this appendix are available to provide more information on using metrics to assess quality and complexity of software.

This bibliography was assembled to provide basic information on the background of the common, industry-accepted metrics. The majority of these metrics were researched from 1970 to 1980. The research into complexity metrics by Halstead, Belady, Lehman, McCabe, Littlewood, Mohanty, Musa, and others did not have the benefits of the COTS tools that today are widely available. The current literature focuses more on cost benefits of using metrics to quantitatively manage software-related activities.

BEIZER, BORIS: Software System Testing and Quality Assurance, Van Norstrand Reinhold Co., 1984.

BOEHM B.W.: Characteristics of Software Quality, TRW North Holland, 1975.

CHIDAMBER, S. and KEMERER, C.: A Metrics Suite for Object Oriented Design, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 20, NO. 6, June, 1994

COLEMAN D., ASH, D., LOWTHER B. and OMAN, P.: Using Metrics to Evaluate Software System Maintainability, IEEE Computer, Aug., 1994.

CONTE S. D., DUNSMORE H.E. and V.Y. SHEN.: SOFTWARE ENGINEERING METRICS AND MODELS, Benjamin/Cummings Publishing Company, 1986.

DeMARCO, T.: Structured Analysis and System Specification, Prentice-Hall, 1978.

DUNN, ROBERT H.: Software Defect Removal, McGraw-Hill Book Company, ISBN 0-07-018313-9, 1984.

GRADY, R. B. and CASWELL, D.: SOFTWARE METRICS: ESTABLISHING A COMPANY-WIDE PROGRAM, PRENTICE-HALL, 1987.

HALSTEAD, M.H.: Elements of Software Science, North Holland, Elsevier , 1977.

HENNELL, M.A., HEDLEY, D., and WOODWARD, M.R.: Experience with Path Analysis and Testing Programs, University of LIVERPOOL Publications.

On Program Analysis, University of LIVERPOOL Publication.

Quantifying the Test Effectiveness of Algol 68 Programs, University of LIVERPOOL Publication.

HENRY, KAFURA and HARRIS: On the Relationships Among Three Software Metrics, ACM Workshop/Symposium on Measurement and Evaluation of Software Quality, 1981 (Proceedings).

HERNDON, M.A.: An Approach Toward the Development of Functional Encoding Model of Short Term Memory During Reading. Journal of Reading Behavior, Vol. X, No. 2, Summer 1978, pp. 141-148.

HERNDON, M.A. and BECKER, W.: Metric Sampling in the Implementation of the Reliability Profile. Proceedings of the Third Minnowbrook Workshop on Software Performance Evaluation, Blue Mountain Lake, NY. 1980.

HERNDON, M.A. and KEENAN, A.P.: Analysis of Error Remediation Expenditures During Validation. Proceedings of the Third International Conference on Software Engineering, pp. 202-206, May 10-11, 1978, Atlanta, GA.

LORAL FEDERAL SYSTEMS: Software Reuse Metrics, Reusability Metrics and Economic Models, April, 1996.

McCABE, T.: A Complexity Measure. IEEE Transactions on Software Engineering, Vol. SE-2, No. 4, pp. 308-320, December 1976.

McCABE,T. and BUTLER, C.W.: Design Complexity Measurement and Testing, Communications of the ACM, Vol. 32, No. 12, pp. 1415-1425, December 1989.

McCALL, J.A.: Factors in Software Quality, General Electric, no. 77C1502, June 1977.

MILLER, G.A.,: The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information, The Psychological Review, Vol. 63, no. 2 (March 1956), pp. 81 - 97.

NEEJMEH, B.A.: NPATH: A Measure of Execution Path Complexity and its Applications, Communications of the ACM, Vol. 31, No. 2, 1988.

MOHANTY, S.N.: Models and Measurements for Quality Assessment of Software Computing Surveys, Vol. 11, No. 3, September 1979.

MUSA, J.D.: The Measurement and Management of Software Reliability, Proceedings of the IEEE, Vol. 68, no. 9 (Sept. 1980), pp. 1131-1143.

MYERS, G.J.: Reliable Software Through Composite Design, Petrucelli/Charter, N.Y., 1975.

MYERS, G.J.: The Art of Software Testing, JOHN WILEY & SONS, 1979.

SCHUTT, D.: On a Hypergraph Oriented Measure for Applied Computer Science, in Proc. COMPCON, pp. 295-296, 1977.

WARD, W.T. and HEWLETT PACKARD: Software Defect Prevention Using McCabe s Complexity Metric, Hewlett-Packard Journal, pp. 65-68, April 1989.

WOODWARD, M.R.: An Investigation into Program Paths and Their Representation, Technique et Science Informatique, Vol. 8, No. 4, 1984.

YOURDON, E.: Structured Walkthroughs, (Third Edition), Yourdon Press, 1985.

This page intentionally left blank.

APPENDIX C: DII SOFTWARE QUALITY ASSESSMENT PROCESS

C.1 Static Analysis Metrics Dictionary

C.1.1 Component Metrics Dictionary

The metrics collected for the COE software quality compliance assessment are listed below and are widely known as indicators of software complexity and quality. Components with metric values exceeding the threshold are viewed as suspect. Additional analysis of source code is required to understand the conditions causing the metric values to exceed the thresholds. Usage of a single metric value in this dictionary is not recommended or valuable. Analysis of the set of metric values is required to provide the information necessary for risk assessment. See Appendix B for additional references.

C.1.1.1 Halstead's Program Length

This is a measure of design, specifically modularity. Modularity is determined during high-level software development activities, and the cost can be high to re-modularize a legacy system.

Threshold Value: 350

Collection Process:

Count the number of unique and total occurrences of operands and operators in a logical unit. These values represent the volume and density of information in the unit.

Example:

X:=+3;	N1 (number of operands) = 10
IF X>B THEN	N2 (number of operators) = 10
A:=B;	n1 (number of unique operands) = 4
B:=X;	n2 (number of unique operators) = 4
ELSE	
A:=X;	Length = N1 + N2 = 20

Informational Value:

If the volume or density of the unit exceeds the threshold values, the unit probably needs to be modularized or simplified.

C.1.1.2 Halstead's Difficulty

This is a measure of how difficult the component was to create. Typically, the harder the development challenge, especially in the "build a little, test a little, and field a little" paradigm, the more risky the integration activities. This is especially true of older, stove-pipe legacy systems.

Threshold Value: 50

Collection Process:

Refer to collection process for length. Difficulty is calculated as the inverse of Program Level. Program Level relates the actual Volume against a logical unit's Optimal Volume. Halstead considered an optimal program to a call to a written procedure.

Optimal Volume is defined as: $V^* = (2+n2) \log2 (2+n2)$

Informational Value:

Measures how difficult the component was to develop.

Typically, the greater the development challenge, the more costly future software-related activities.

C.1.1.3 Cyclomatic Number

A measure of the number of testable paths in a module. Values exceeding the threshold usually signal a too informal implementation of control logic.

Threshold Value: 15

Collection Process:

Develop a directed graph to represent the control structure of the logical unit. The control graph is developed by representing the code as a series of nodes, conditions, ends of conditions, and edges. A node is a statement or block of sequential statements, represented on the graph as a circle. A condition, represented as a diamond, has multiple branches. The end of the condition represents the end of the branches. An edge is the flow of control between nodes.

Informational Value:

Measures the difficulty and feasibility of the testing.

Control structures with large values of cyclomatic number may contain residual errors in the control logic.

C.1.1.4 Essential Complexity

A measure of the structure of testable paths in a component. Values exceeding the threshold usually signal an unstructured implementation of control logic.

Threshold Value: 7

Collection Process:

Develop a directed graph to represent the control structure of the logical unit. This graph should contain only the four basic and simple structured constructs: Sequence, If-Then (or If-Then-Else), Loops (While-Do and Do-Until), Case. Replace occurrences of unstructured constructs by one of the four simple structured constructs.

Informational Value:

Measures the difficulty and feasibility of the testing as the presence of unstructured constructs increases effort of testing as well as future code modifications.

Control structures with large values of essential complexity may need process improvement.

C.1.1.5 Design Complexity

This is a measure of the control of flow implemented by the design. This metric indicates the degree of difficulty of integration testing.

Threshold Value: 10

Collection Process:

Develop a reduced directed graph to represent the control structure of the logical unit. This graph should contain only decisions and nodes impacting calls to subordinate logical units.

Informational Value:

Measures the minimum number of integration tests.

Control structures with large values of essential complexity may need their design simplified.

C.1.1.6 Source Lines of Code

The physical length of software is known to be a factor in determining the difficulty of changing legacy systems.

Threshold Value: 70

Collection Process:

Count the number of complete executable statements. Comments are not included.

Informational Value:

Measures the difficulty of understanding the functionality of the software.

Logical units with large values of SLOC may need design simplification and process improvement.

C.1.1.7 Control Density

A measure of the "richness" of decision nodes in a module. A high density module will be more difficult to maintain than a less dense one.

Threshold Value: .33

Collection Process:

Count the number of control statements and the number complete executable statements not including comments. Control density is the ratio of the number of control statements to the total number of statements.

Informational Value:

Measures the difficulty of understanding the functionality of the software.

Logical units with large values of control density may need design and control structure simplification and process improvement.

C.1.1.8 Maximum Number of Levels

Since the 1950's, Cognitive Sciences studies have shown that groups that contain more than seven pieces of information as increasingly harder for people to understand in problem solving.

Threshold Value: 7

Collection Process:

Count the number of If ...Then or If ...Then..Elses in a nest.

Informational Value:

Measures the difficulty of understanding the control logic of the software.

Logical units with a large number of nested levels may need implementation simplification and process improvement.

C.1.1.9 Number of Branching Nodes

Higher values indicate possible use of “gotos” and/or abnormal exits from control structures such as loops. This problem is an indicator of unstructured design and increases the testing difficulty. For the components written in C, a higher value may be permitted due to the legitimate use of C “breaks” in a “switch” statement.

Threshold Value: 7

Collection Process:

Refer to the essential complexity collection process.

Informational Value:

Increases the difficulty of testing the control logic of the software.

C.1.1.10 Number of Input/Output Nodes

Programming practices today state there should be one way into a module and one way out.

Threshold Value: 2

Collection Process:

Refer to the essential complexity collection process.

Informational Value:

Measures the difficulty of testing the control logic of the software.

Logical units a large number of input/output nodes may need implementation simplification and process improvement.

C.1.2 Call Graph Metrics Dictionary

The metrics discussed in this subsection are collected to assess the complexity of the implementation of a specific call graph architecture. The information provided by this set of metrics assesses the data and control flow complexity for the call graph similar to an assessment within a component. The metrics focus on the modularity of the design, complexity of the implementation, amount of understanding needed to modify the code, projected testing effort, and

adherence to industry-accepted good programming practices. These metrics are collected by the COTS tool and would be impractical to collect manually.

C.1.2.1 Hierarchical Complexity

A measure of the average number of components on a level. The greater the number of components, the greater the number of test cases required to test effectively.

Threshold Value: 5

Collection Process:

Prepare the call graph for each root node in the hierarchy.

Informational Value:

Measures the difficulty of testing the software.

Applications with many component levels may contain a large number of untested paths that tend to be unexpectedly executed. This problem can plague client/server applications.

C.1.2.2 Structural Complexity

The average number of calls per component in the call graph. The larger the value of this metric, the greater the difficulty in testing and debugging the code.

Threshold Value: 3

Collection Process:

Prepare the call graph for each root node in the hierarchy.

Informational Value:

Measures the difficulty of testing the software.

Applications with large values of component levels may contain a large number of untested paths that tend to be unexpectedly executed. This problem can plague client/server applications.

C.1.2.3 Average Paths

The average number of paths in the call tree. As the value of this metric increases, the testing and debugging becomes more difficult.

Threshold Value: 2

Collection Process:

Prepare the call graph for each root node in the hierarchy.

Informational Value:

Measures the difficulty of testing the software.

Applications with large values of component levels may contain a large number of untested paths that tend to be unexpectedly executed. This problem can plague client/server applications.

C.1.2.4 Number of Levels

The number of levels in the call tree. As the depth of a tree increases, the testing challenge increases.

Threshold Value: 9

Collection Process:

Prepare the call graph for each root node in the hierarchy.

Informational Value:

Measures the difficulty of testing the software.

Applications with large values of component levels may contain a large number of untested paths that tend to be unexpectedly executed. This problem can plague client/server applications.

C.1.2.5 Entropy

A measure of the orderliness in the execution of the components in a call graph. The less order in the execution paths, the harder to test and debug.

Threshold Value: 3

Collection Process:

Prepare the call graph for each root node in the hierarchy.

Informational Value:

Measures the difficulty of adequately testing the software. Understanding the flow of data becomes increasing difficult.

Applications with large values of component levels may contain a large number of untested paths that tend to be unexpectedly executed. This problem can plague client/server applications.

C.2 Quality Criteria Profile

The criterion formulas that define the members of the Quality Criteria Profile each use a specific set of metrics. After the metrics in the dictionary are collected from each COE candidate, a criterion is calculated by first assigning a value of one to each metric that falls within the acceptable range and zero if otherwise. This value is multiplied by the weight assigned to that metric in the formula. The total of the weighted values divided by the maximum possible value gives the percentage used to determine the criterion category. The QCP for the components provides information describing Maintainability, Correctness, and Reliability. Currently, there is a QCP computed for the components and an architecture metric value computed for the call graphs.

C.2.1 QCP Formulas and Risk Assignment Intervals

The information provided by the QCP risk interval assignments provides DII with a non-intrusive assessment of COE Software Quality compliance for each candidate.

Maintainability: Effort required to modify the applications software.

Correctness: Extent of simplicity and structure in the application software logic.

Reliability: Extent to which the applications software performs functions with required precision and robustness.

QCP CRITERIA: MAINTAINABILITY

MAINTAINABILITY = 3 * Program Length + Number of Statements + Maximum Levels + 2 * Cyclomatic Number + Number of Branching Nodes.

Risk Interval	Risk Reduction	Score Range
None	Accept as is	80-100
Low	Improve documentation	50-80
Medium	Re-implement control logic	30-50
High	Re-design	0-30

QCP CRITERIA: CORRECTNESS

CORRECTNESS = Halstead's Difficulty + Number of Statements + 2 * Cyclomatic Number

Risk Interval	Risk Reduction	Score Range
None	Accept as is	75-100
Low	Improve testing	50-75
Medium	Re-implement control logic	25-50
High	Re-design	0-25

QCP CRITERIA: RELIABILITY

RELIABILITY = Program Length + 2 * Maximum Levels + 3 * Cyclomatic Number + Number of Branching Nodes + Number of Input and Output Nodes + Control Density

Risk Interval	Risk Reduction	Score Range
None	Accept as is	75-100
Low	Improve testing	50-75
Medium	Re-implement control logic	25-50
High	Re-design	0-25

Together the three Criteria define the QCP for the set of components in a COE candidate. For each Criteria, these computed values are used to assign a component membership in a risk interval with a recommended course of action. The lower and upper ranges associated with each category are set by the analyst based on past experiences and good software engineering practices.

The metrics process for collection and analysis of the call graph metrics is similar to the process used to generate the QCP.

ARCHITECTURE

ARCHITECTURE = Normalized value of (Hierarchical Complexity + Structural Complexity + Average Paths + Levels + Entropy)/5

Category	Score Range
ACCEPT	60-100
REDESIGN	0-59

C.3 Minimum Set Assessment Process

The steps used in the MS assessment process are:

Step 1: For each component in the candidate CSCI/CSC, risk population information is collected for each Criteria in the QCP.

Step 2: The high risk components are listed for each QCP Criteria.

Step 3: The common components falling in the high-risk population interval for all three Criteria in the QCP are identified as the Minimum Set.

C.4 Emerging Risk Assessment Process

The steps in the Emerging Risk analysis are:

Step 1: For each element in the component set, a risk analysis is performed for each Criteria in the QCP.

Step 2: The number of elements that fall into each of the three intervals are counted: Low, Medium, and High.

Step 3: A calculation is performed using the values calculated in Step 2 to determine the Expansion Factor.

C.5 Complexity and Quality Analysis Assessment Process

The steps in the Complexity and Quality Analysis are.

Step 1: For each element in the component set, a risk analysis is performed for each of the 4 metrics.

Step 2: The medium and high risk elements are listed for each metric.

Step 3: A count is made of elements in the medium and high risk intervals and the percentage of the total number of components calculated.